PROJET-JAVA

Java project by Peter and Younes

Younes was charged with the design of the graphical user interface and Peter handled the synchronization logic in local and network and we both worked on the UML.

Project which consists in synchronizing two folders in Java, whether it is in local or in network, without the help of extern API.

The synchronization is possible on a single computer in local, or on network in local, using two computers. The synchronization works both ways: if the folder A is modified, then the folder B is also modified, and the way around.

To send data through sockets, we first serialize the data in a String buffer, then the receiver deserialize the data, using the separators we have chosen "||". The receiver receives in order the type of the file : 1 if it's a folder, 0 if it's a file, then the relative path of the file, and finally the content of the file, if it's one.

There is a lot of recursion in the project, as we can't directly send a folder with its file, or copy a whole folder. To do so, we first create the folder, then call the method recursively, to ensure that every subfolders are sent.

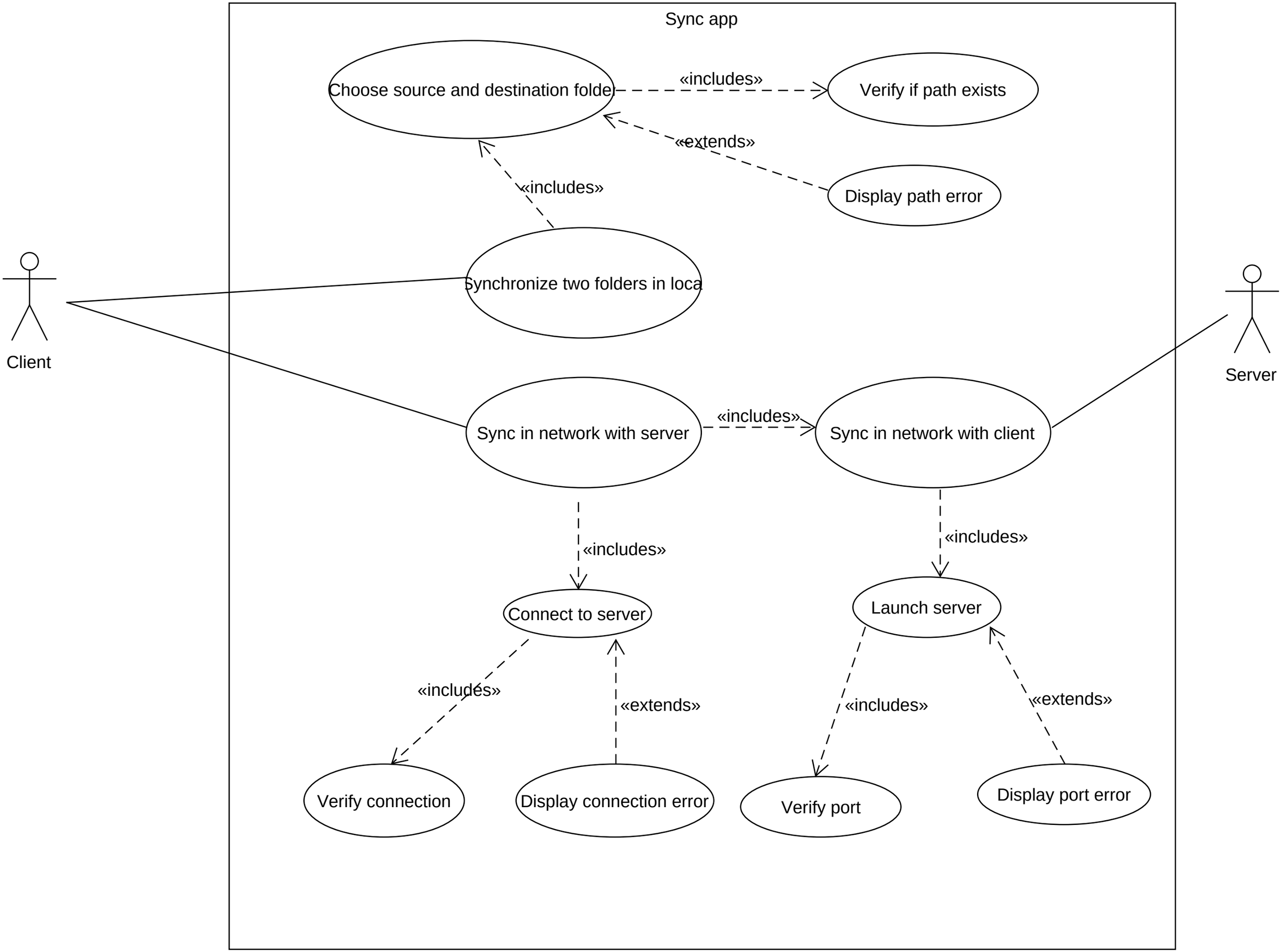
Execution : java -jar peter_younes.jar or open the jar file.

To make it work in local, on a single computer, the user must choose the source folder he wants to synchronize, and the destination folder, where it needs to be synchronized, then press the Synchronize button.

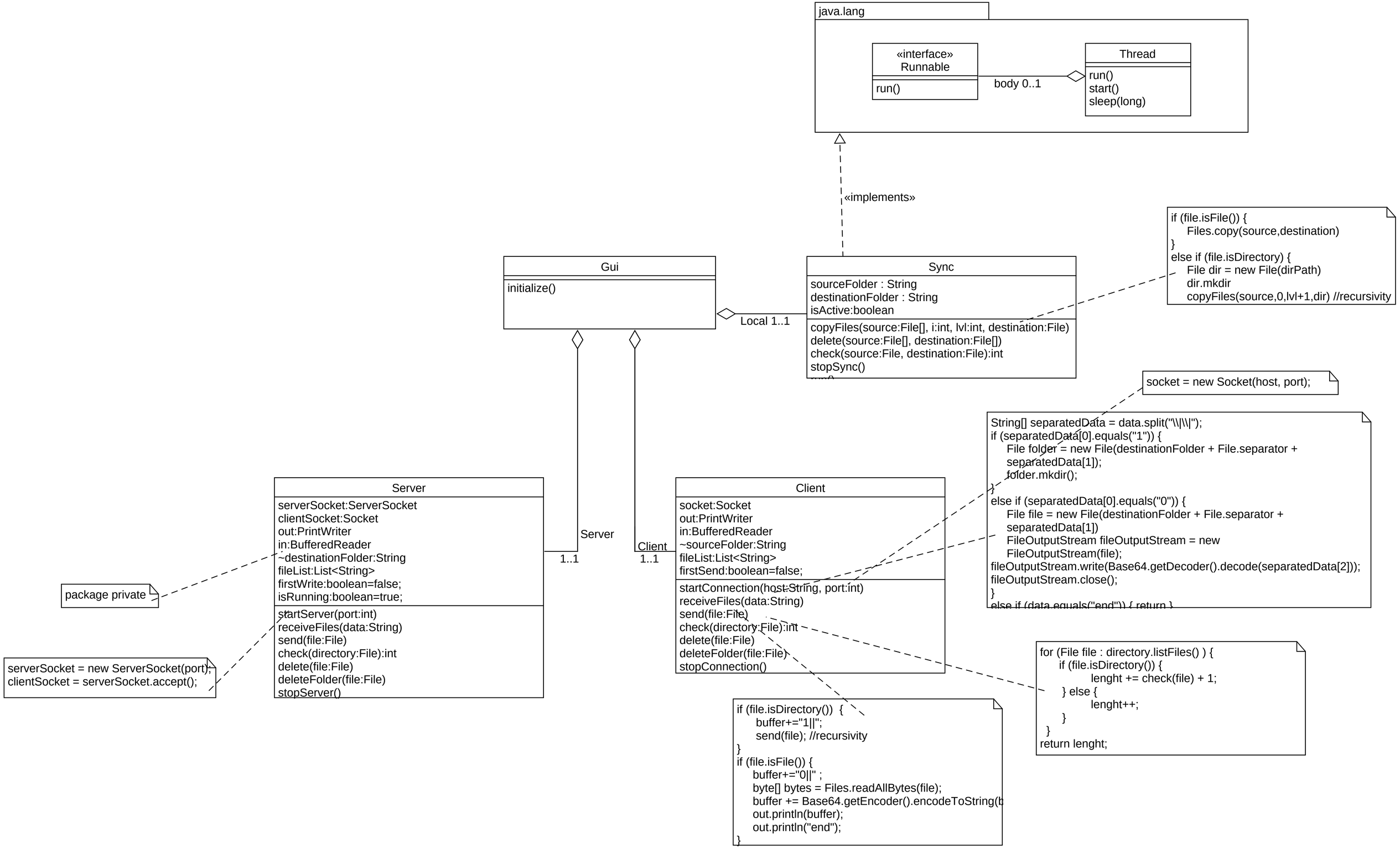
To make it work in network, the server must choose a folder where he will receive the files, and choose a port, then launch the server. Only then, the client can choose the folder he wishes to send, choose the same port as the server, and connect to it.

Extra functionality : The graphical user interface is responsive and interactive, it changes according to what the user want to do (synchronize in network or local)

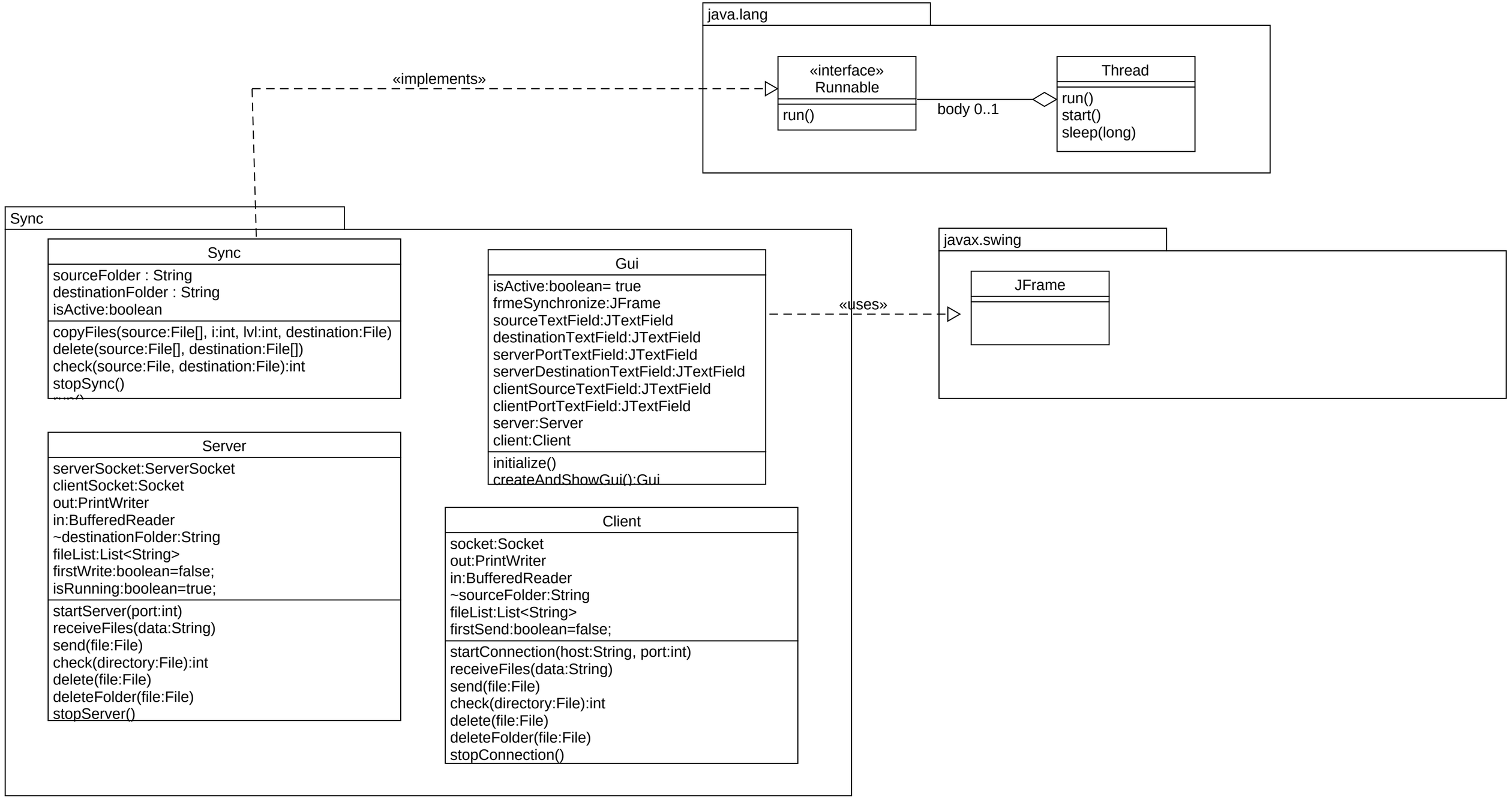
Use case


Sync app
Choose source and destination folde
«includes»
Verify if path exists
«extends»
«includes»
Display path error
Synchronize two folders in loca
Client
Server
Sync in network with server
«includes»
Sync in network with client
«includes»
«includes»
Connect to server
Launch server
«includes»
«extends»
«includes»
«extends»
Verify connection
Display connection error
Verify port
Display port error

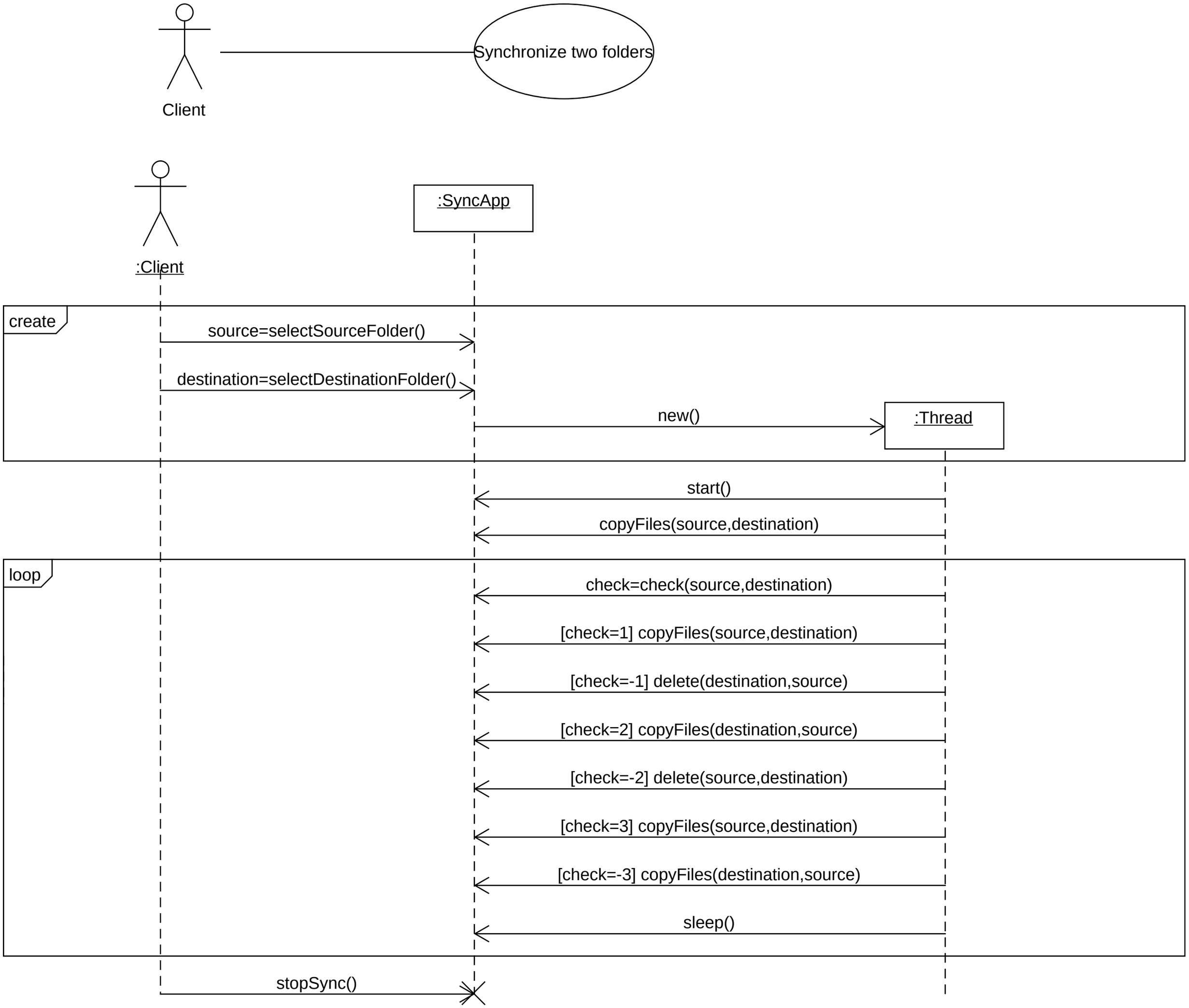
Class Diagram

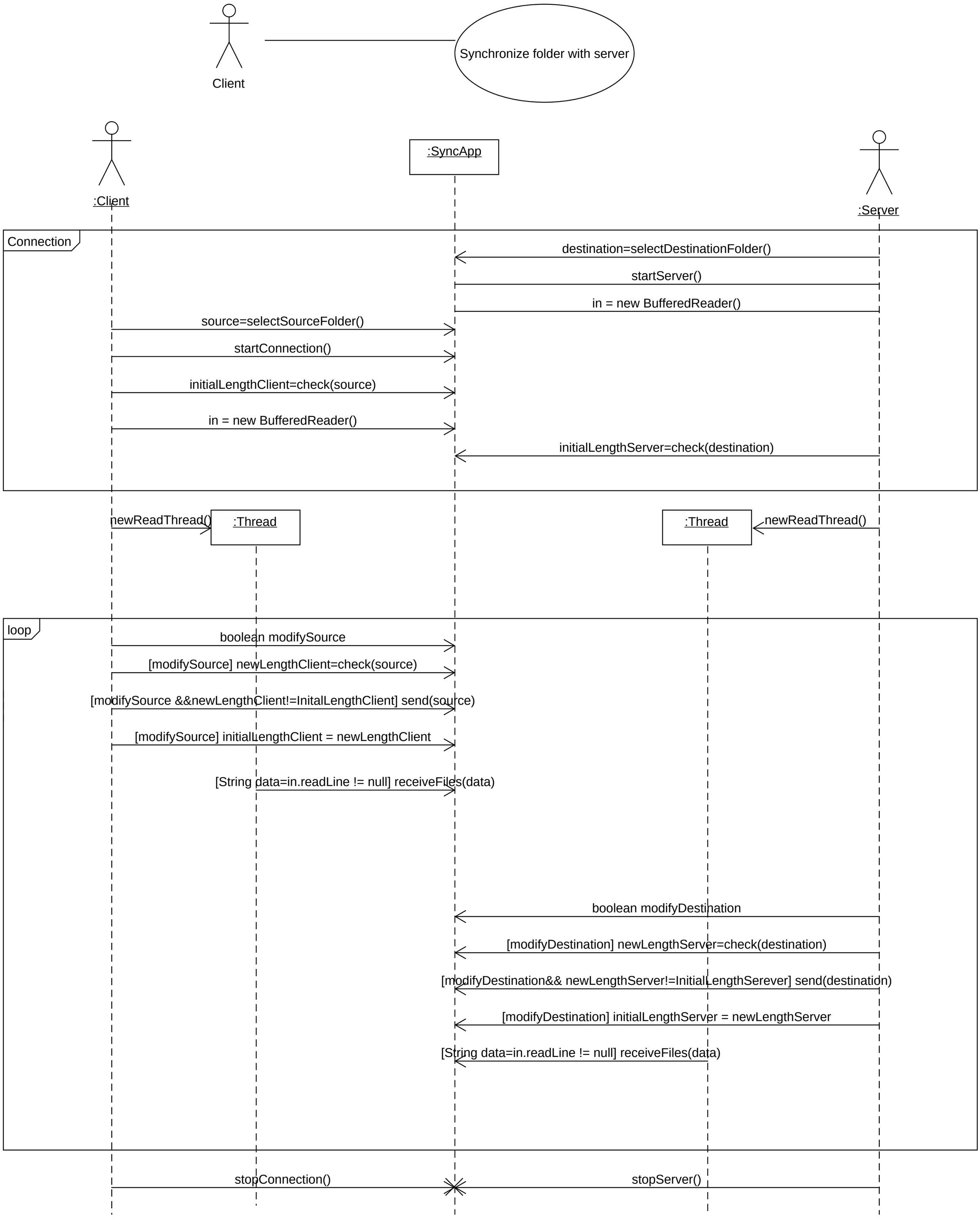

java.lang
«interface»
Runnable
run()
body 0..1
Thread
run()
start()
sleep(long)
«implements»
if (file.isFile()) {
Files.copy(source,destination)
}
else if (file.isDirectory) {
File dir = new File(dirPath)
dir.mkdir
copyFiles(source,0,lvl+1,dir) //recursivity
Gui
initialize()
Local 1..1
Sync
sourceFolder : String
destinationFolder : String
isActive:boolean
copyFiles(source:File[], i:int, lvl:int, destination:File)
delete(source:File[], destination:File[])
check(source:File, destination:File):int
stopSync()
socket = new Socket(host, port);
String[] separatedData = data.split("\\|\\|");
if (separatedData[0].equals("1")) {
File folder = new File(destinationFolder + File.separator +
separatedData[1]);
folder.mkdir();
}
else if (separatedData[0].equals("0")) {
File file = new File(destinationFolder + File.separator +
separatedData[1])
FileOutputStream fileOutputStream = new
FileOutputStream(file);
fileOutputStream.write(Base64.getDecoder().decode(separatedData[2]));
fileOutputStream.close();
}
Server
serverSocket:ServerSocket
clientSocket:Socket
out:PrintWriter
in:BufferedReader
~destinationFolder:String
fileList:List<String>
firstWrite:boolean=false;
isRunning:boolean=true;
startServer(port:int)
receiveFiles(data:String)
send(file:File)
check(directory:File):int
delete(file:File)
deleteFolder(file:File)
stopServer()
Server
1..1
Client
1..1
Client
socket:Socket
out:PrintWriter
in:BufferedReader
~sourceFolder:String
fileList:List<String>
firstSend:boolean=false;
startConnection(host:String, port:int)
receiveFiles(data:String)
send(file:File)
check(directory:File):int
delete(file:File)
deleteFolder(file:File)
stopConnection()
package private
serverSocket = new ServerSocket(port);
clientSocket = serverSocket.accept();
if (file.isDirectory()) {
buffer+="1||";
send(file); //recursivity
}
if (file.isFile()) {
buffer+="0||" ;
byte[] bytes = Files.readAllBytes(file);
buffer += Base64.getEncoder().encodeToString(b
out.println(buffer);
out.println("end");
}
for (File file : directory.listFiles()) {
if (file.isDirectory()) {
lenght += check(file) + 1;
} else {
lenght++;
}
}
return lenght;

Package diagram

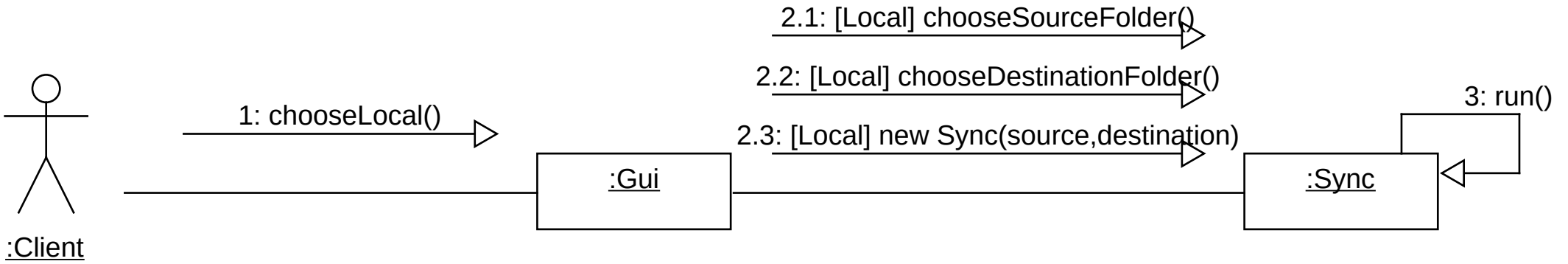

java.lang
«interface»
Runnable
run()
Thread
run()
start()
sleep(long)
«implements»
body 0..1
Sync
Sync
sourceFolder : String
destinationFolder : String
isActive:boolean
copyFiles(source:File[], i:int, lvl:int, destination:File)
delete(source:File[], destination:File[])
check(source:File, destination:File):int
stopSync()
Gui
isActive:boolean= true
frmeSynchronize:JFrame
sourceTextField:JTextField
destinationTextField:JTextField
serverPortTextField:JTextField
serverDestinationTextField:JTextField
clientSourceTextField:JTextField
clientPortTextField:JTextField
server:Server
client:Client
initialize()
«uses»
javax.swing
JFrame
Server
serverSocket:ServerSocket
clientSocket:Socket
out:PrintWriter
in:BufferedReader
~destinationFolder:String
fileList:List<String>
firstWrite:boolean=false;
isRunning:boolean=true;
startServer(port:int)
receiveFiles(data:String)
send(file:File)
check(directory:File):int
delete(file:File)
deleteFolder(file:File)
stopServer()
Client
socket:Socket
out:PrintWriter
in:BufferedReader
~sourceFolder:String
fileList:List<String>
firstSend:boolean=false;
startConnection(host:String, port:int)
receiveFiles(data:String)
send(file:File)
check(directory:File):int
delete(file:File)
deleteFolder(file:File)
stopConnection()

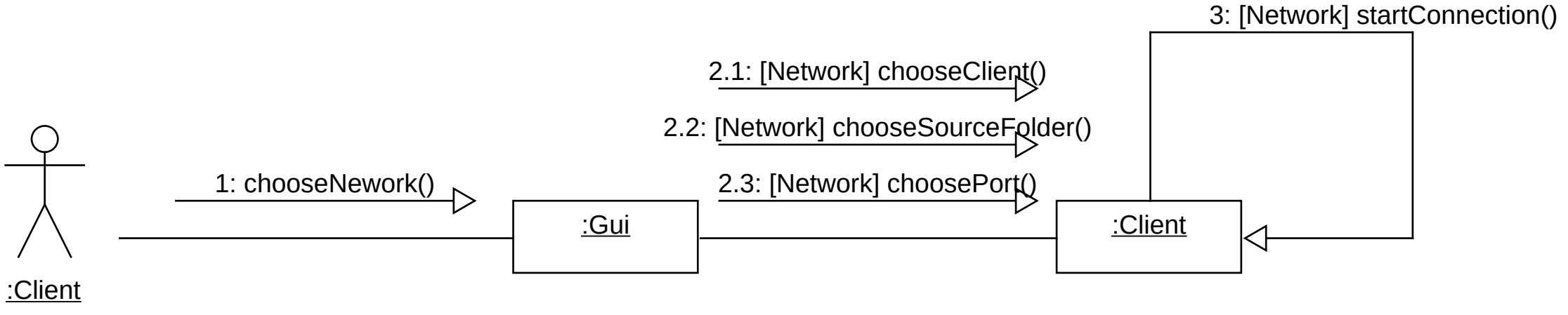
Sequence Diagram


Client
Synchronize two folders
:Client
:SyncApp
create
source=selectSourceFolder()
destination=selectDestinationFolder()
new()
:Thread
start()
copyFiles(source,destination)
loop
check=check(source,destination)
[check=1] copyFiles(source,destination)
[check=-1] delete(destination,source)
[check=2] copyFiles(destination,source)
[check=-2] delete(source,destination)
[check=3] copyFiles(source,destination)
[check=-3] copyFiles(destination,source)
sleep()
stopSync()

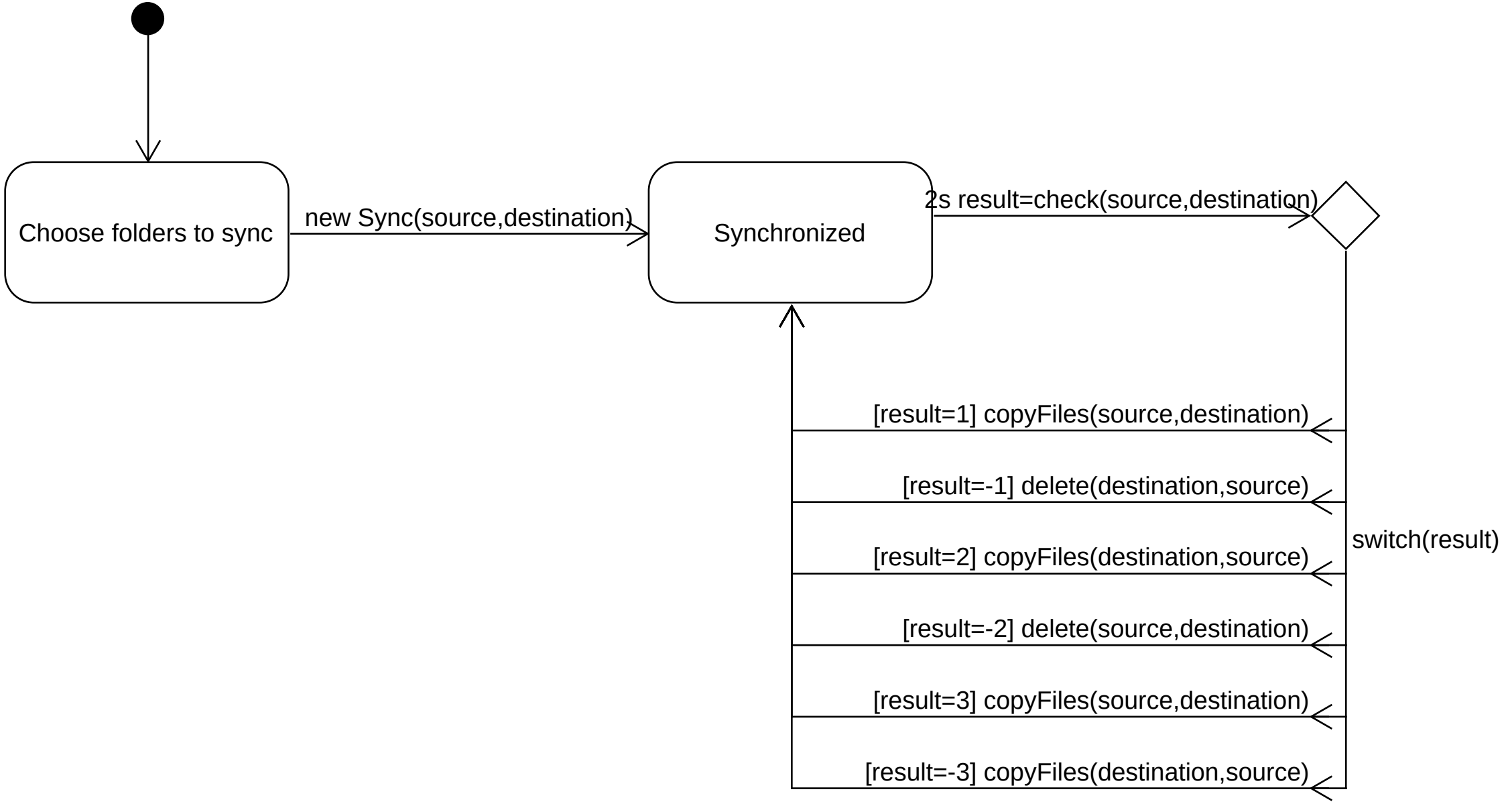

Client
Synchronize folder with server
:Client
:SyncApp
:Server
Connection
destination=selectDestinationFolder()
startServer()
in = new BufferedReader()
source=selectSourceFolder()
startConnection()
initialLengthClient=check(source)
in = new BufferedReader()
initialLengthServer=check(destination)
newReadThread()
:Thread
:Thread
newReadThread()
loop
boolean modifySource
[modifySource] newLengthClient=check(source)
[modifySource &&newLengthClient!=InitalLengthClient] send(source)
[modifySource] initialLengthClient = newLengthClient
[String data=in.readLine != null] receiveFiles(data)
boolean modifyDestination
[modifyDestination] newLengthServer=check(destination)
[modifyDestination&& newLengthServer!=InitialLengthSerever] send(destination)
[modifyDestination] initialLengthServer = newLengthServer
[String data=in.readLine != null] receiveFiles(data)
stopConnection()
stopServer()

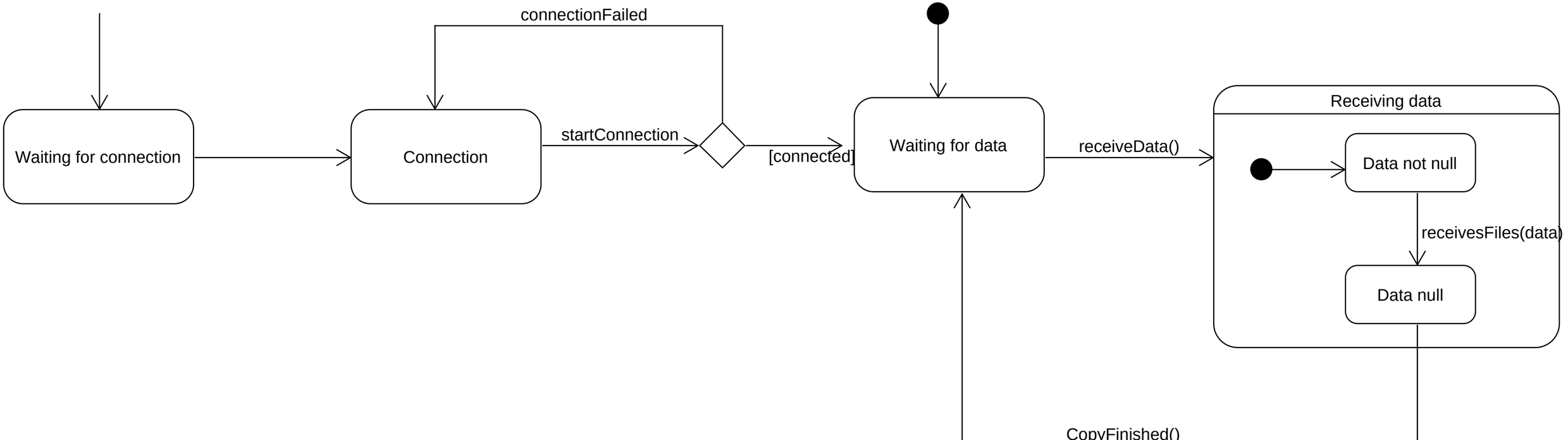
Communication diagram

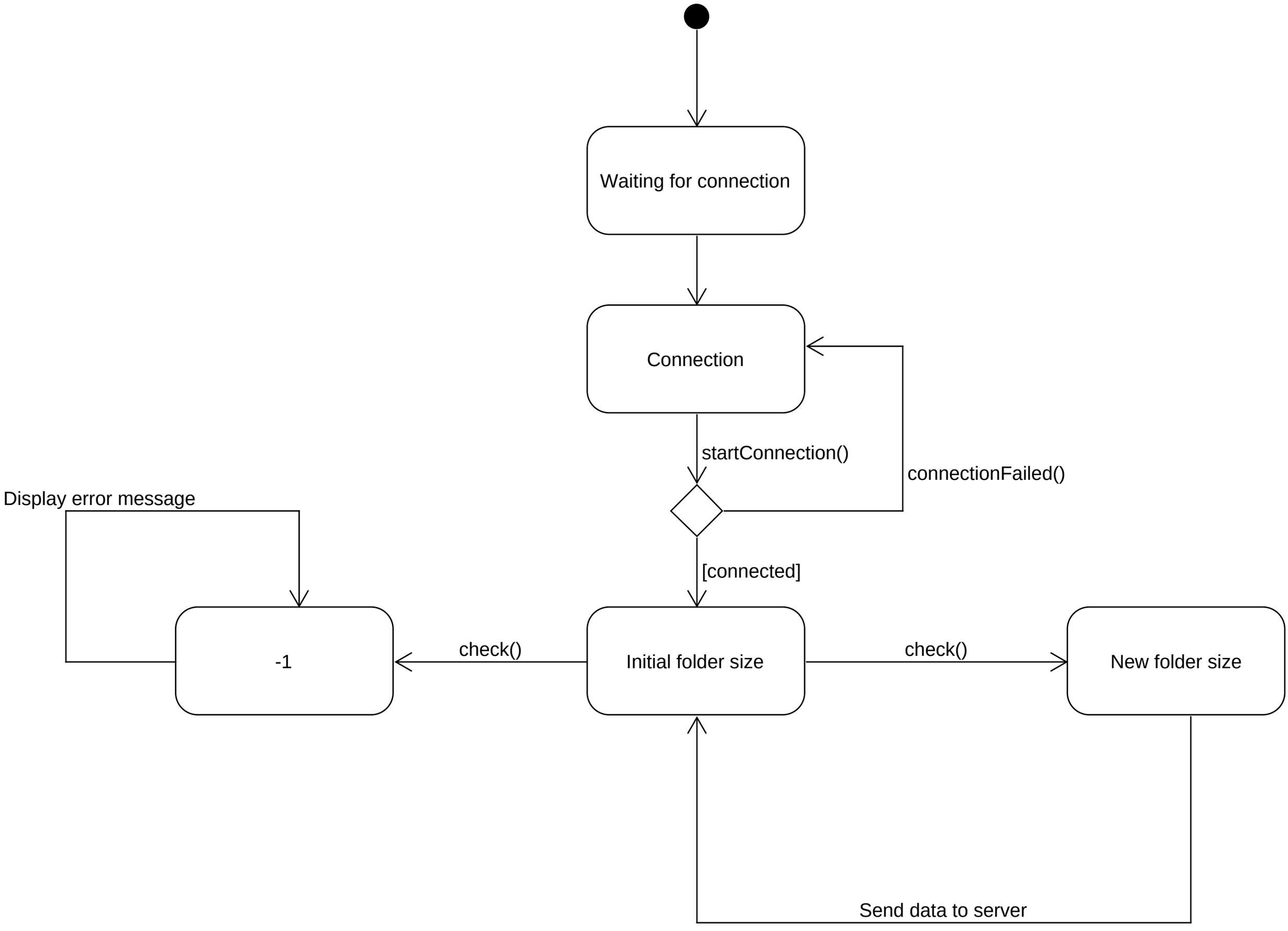

2.1: [Local] chooseSourceFolder()
2.2: [Local] chooseDestinationFolder()
3: run()
1: chooseLocal()
2.3: [Local] new Sync(source,destination)
:Gui
:Sync
:Client


3: [Network] startConnection()
2.1: [Network] chooseClient()
2.2: [Network] chooseSourceFolder()
2.3: [Network] choosePort()
1: chooseNework()
:Gui
:Client
:Client

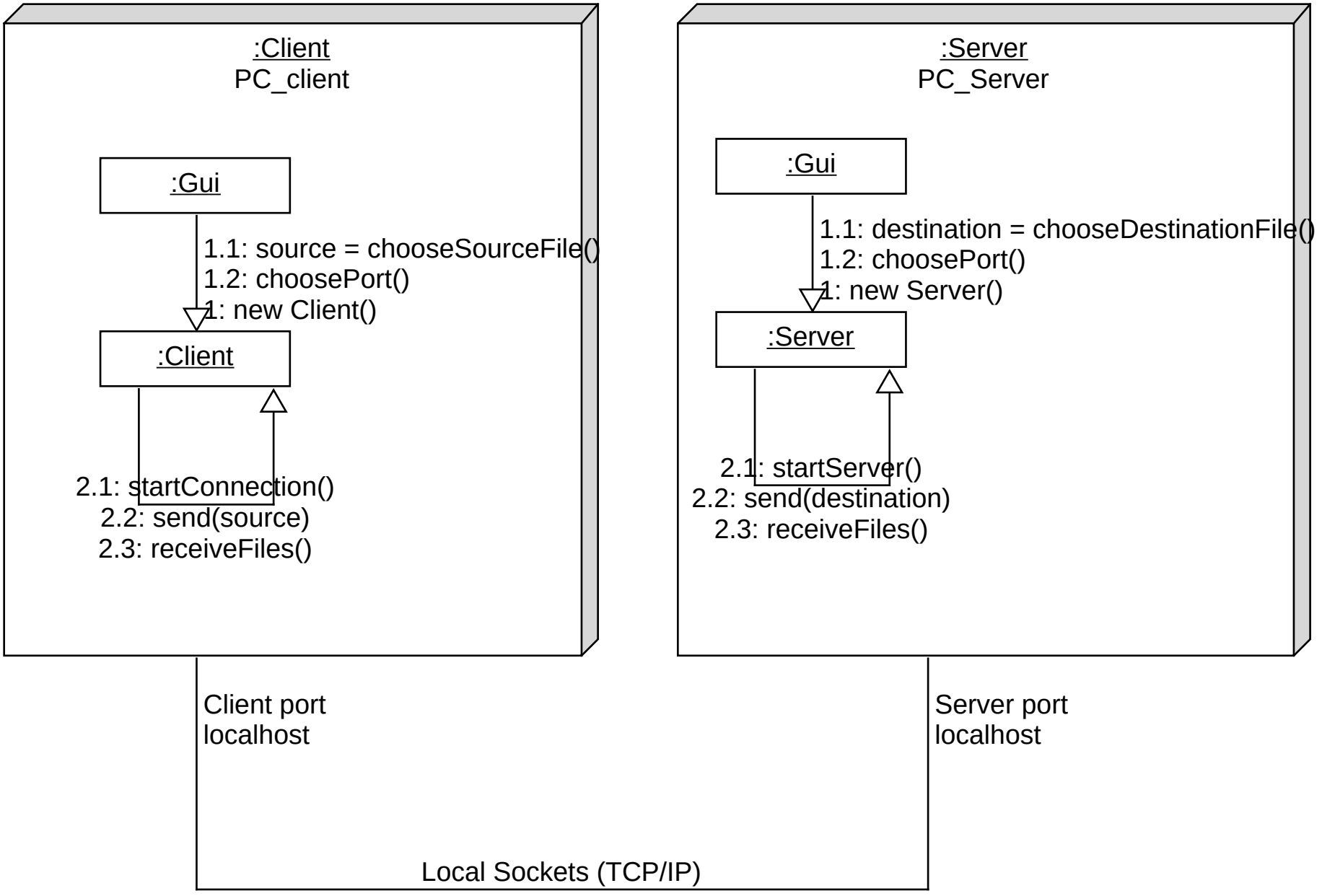
State diagram


Choose folders to sync
new Sync(source,destination)
Synchronized
2s result=check(source,destination)
[result=1] copyFiles(source,destination)
[result=-1] delete(destination,source)
[result=2] copyFiles(destination,source)
[result=-2] delete(source,destination)
[result=3] copyFiles(source,destination)
[result=-3] copyFiles(destination,source)
switch(result)

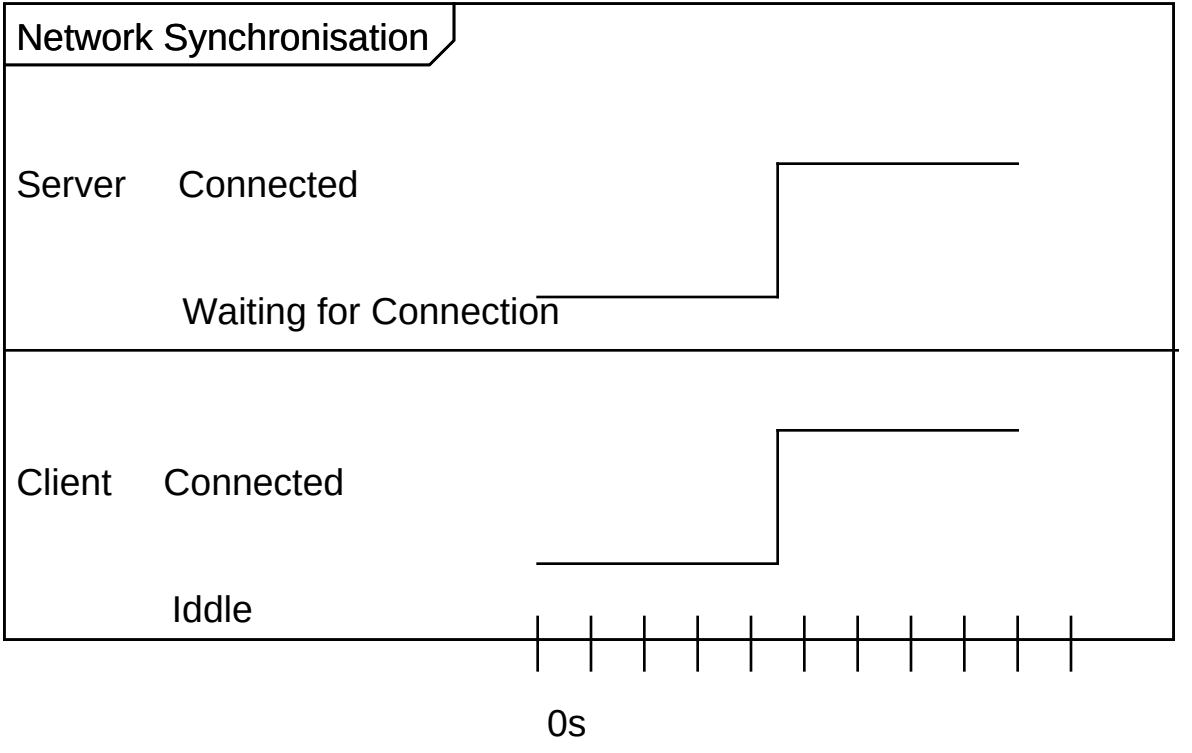

Waiting for connection
Connection
connectionFailed
startConnection
[connected]
Waiting for data
receiveData()
Receiving data
Data not null
receivesFiles(data)
Data null
CopyFinished()


Waiting for connection
Connection
startConnection()
connectionFailed()
[connected]
Initial folder size
check()
-1
Display error message
check()
New folder size
Send data to server

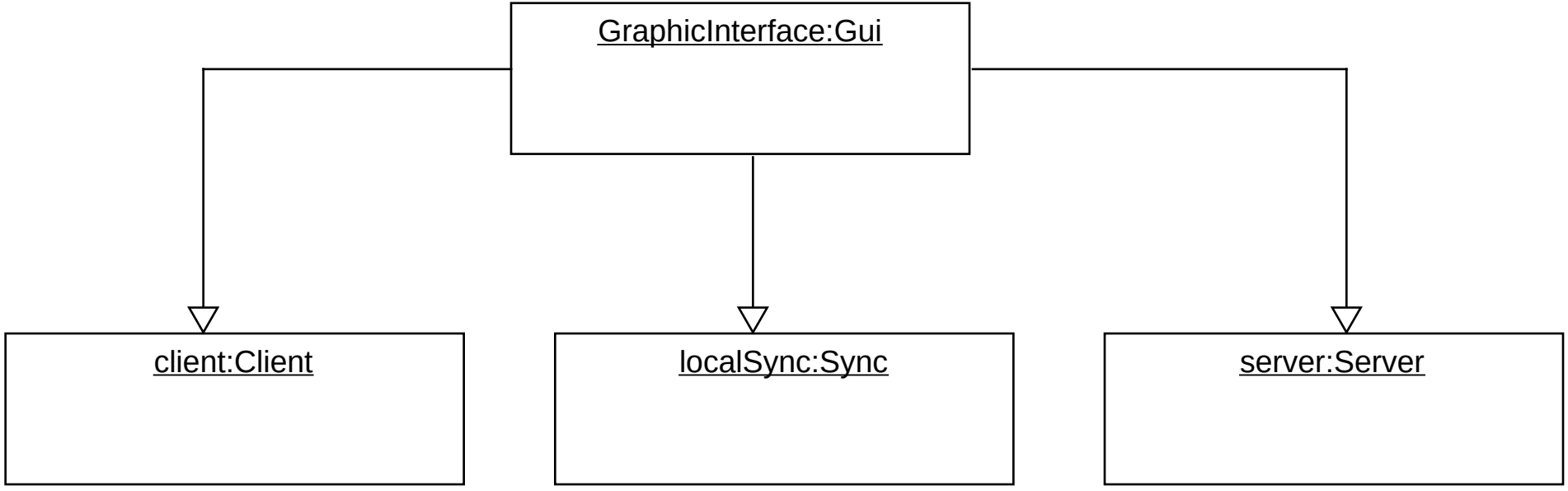
Deployment diagram


:Client
PC_client
:Gui
1.1: source = chooseSourceFile()
1.2: choosePort()
1: new Client()
:Client
2.1: startConnection()
2.2: send(source)
2.3: receiveFiles()
:Server
PC_Server
:Gui
1.1: destination = chooseDestinationFile()
1.2: choosePort()
1: new Server()
:Server
2.1: startServer()
2.2: send(destination)
2.3: receiveFiles()
Client port
localhost
Server port
localhost
Local Sockets (TCP/IP)

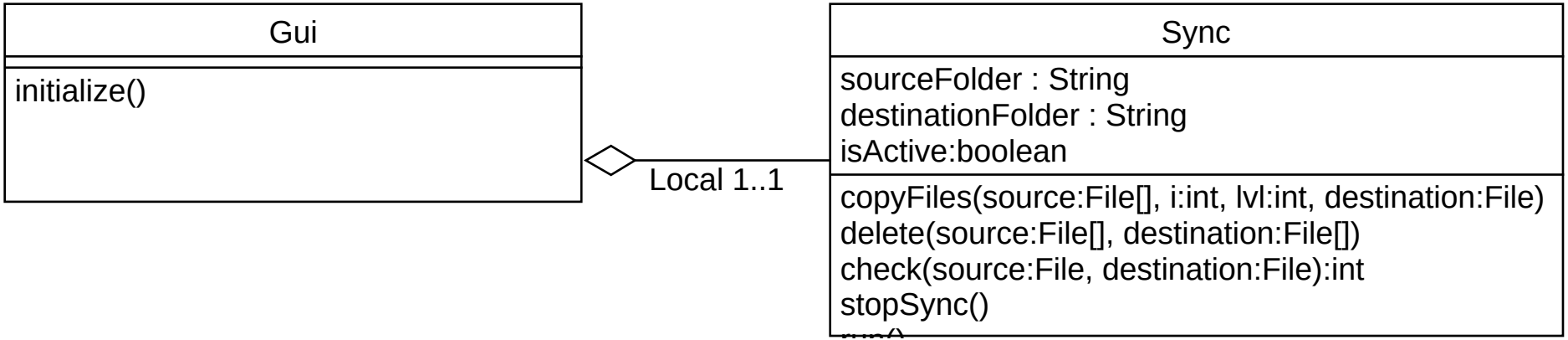
Timing diagram


Network Synchronisation
Server
Connected
Waiting for Connection
Client
Connected
Iddle
0s

Object diagram


GraphicInterface:Gui
client:Client
localSync:Sync
server:Server

OCL CONSTRAINTS

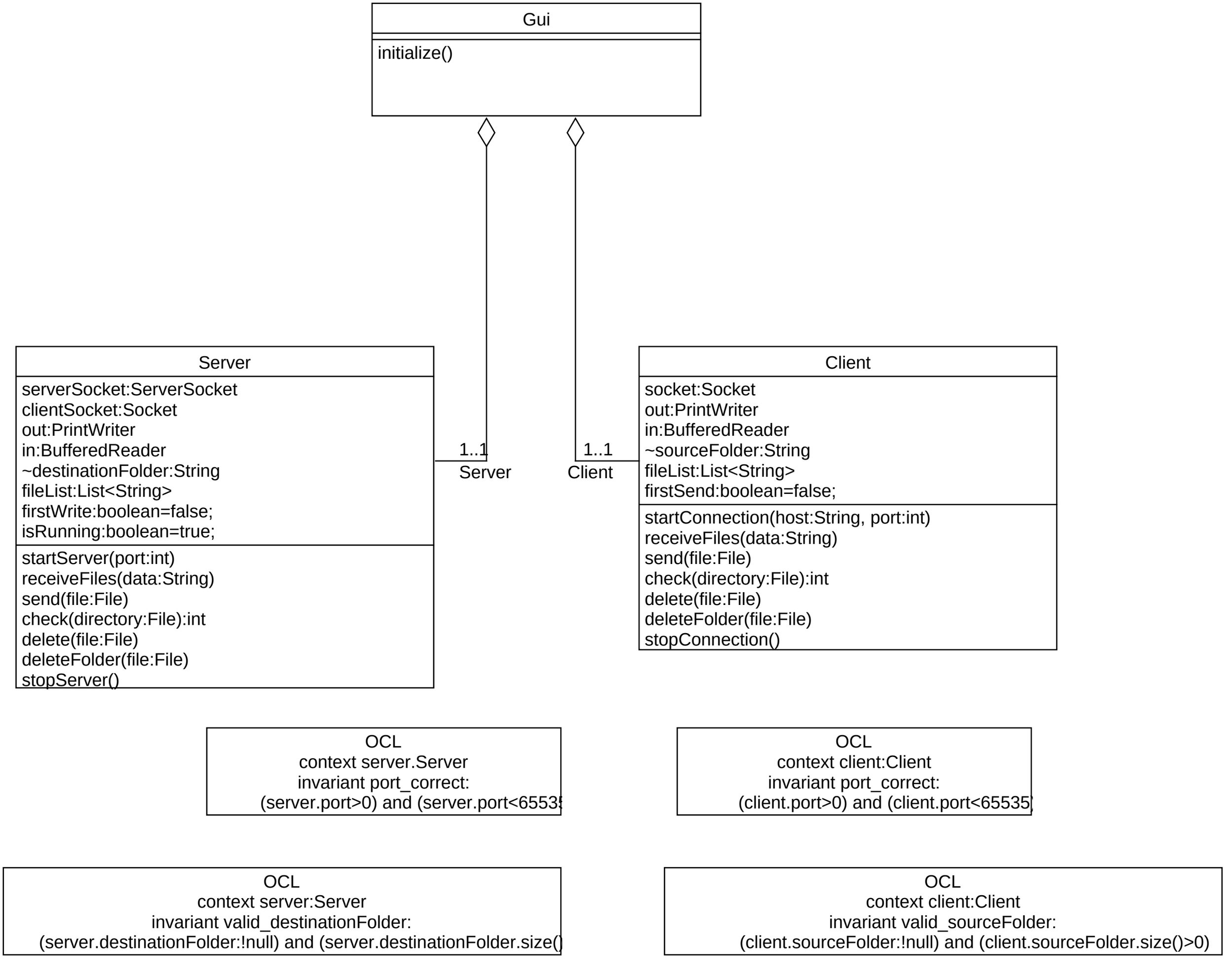


OCL

context sync.Sync

invariant valid_sourceFolder:

(sync.sourceFolder!=sync.destinationFolder)


Gui
initialize()
Server
serverSocket:ServerSocket
clientSocket:Socket
out:PrintWriter
in:BufferedReader
~destinationFolder:String
fileList:List<String>
firstWrite:boolean=false;
isRunning:boolean=true;
startServer(port:int)
receiveFiles(data:String)
send(file:File)
check(directory:File):int
delete(file:File)
deleteFolder(file:File)
stopServer()
1..1
Server
1..1
Client
Client
socket:Socket
out:PrintWriter
in:BufferedReader
~sourceFolder:String
fileList:List<String>
firstSend:boolean=false;
startConnection(host:String, port:int)
receiveFiles(data:String)
send(file:File)
check(directory:File):int
delete(file:File)
deleteFolder(file:File)
stopConnection()
OCL
context server.Server
invariant port_correct:
(server.port>0) and (server.port<65535
OCL
context client:Client
invariant port_correct:
(client.port>0) and (client.port<65535)
OCL
context server:Server
invariant valid_destinationFolder:
(server.destinationFolder:!null) and (server.destinationFolder.size()
OCL
context client:Client
invariant valid_sourceFolder:
(client.sourceFolder:!null) and (client.sourceFolder.size()>0)